MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

A FAST BOUNDARY BASED THINNING ALGORITHM

Y. Zhu, L. D. Seneviratne and S. W. E. Earles

Department of Mechanical Engineering, King's College London,
Strand, London, WC2R 2LS, U. K.

Abstract

Thinning algorithms can be classified into two general types: sequential and parallel algorithms. Most of them peel off the object boundaries until the objects have been reduced to thin lines. The process is performed iteratively and needs a number of iterations (approximately equal to half of the maximum line width of the object). Several boundary based algorithms which belong to the sequential method have been proposed, but they have limitations. A new boundary based algorithm is presented. Experimental results are used to compare this new algorithm to other sequential algorithms and their relative performances are assessed.

I. INTRODUCTION

Thinning algorithms can be classified into two general types: sequential [1-5] and parallel [11-22] algorithms. A common character of these algorithms is that they are pixel-based. They check the condition of small local neighbourhoods around the pixel (usually a 3×3 mask or larger, 8 neighbours or more) to find out the deletable boundary pixels of the object, whose removal does not locally disconnect their neighbourhoods or cause excessive erosion. The process is performed iteratively. In each iteration, every image pixel or boundary pixels (for contour following methods) are inspected and single-pixel wide boundaries, that are not required to maintain connectivity or endlines, are erased. The number of iterations is approximately equal to half of the maximum line width of the object. O'Gorman [23] enlarged the size of mask to k×k (k≥3) and a center core of (k-2)×(k-2) pixels is erased if the deletion criteria are satisfied. Because thicker layers are peeled from the boundary of the object, fewer iterations are required to reach the thinned result. For a large k, this is often at the cost of an increase in the coarseness of the result.

Another type of sequential methods are based on contours to calculate an approximate median line [7-9]. Shapiro *et al.* [9] exploited the trapezoid formed by two pairs of boundary points along the two boundary segments. If the diagonals of the trapezoid are nearly equal, then the point of intersection of the diagonals represents a point on the median line. Their algorithm can deal with ribbon-like or simple tree-like objects. Even for these types of shapes, there are problems in deciding what constitutes a legitimate skeleton for the ambiguous regions. Using the vectorized borders of the objects as input data, Martinez-Perez *et al.* [8] tried to find a pair of "opposite" points by throwing a projecting line from the vertex to the facing segment and generate sequentially the median line, by adding the mid-point between the vertex and the intersection. Their algorithm is intended for "elongated" objects, with a clearly defined longitudinal direction, and an approximate constant thickness. It is difficult to deal with the shape when a small hole exists in the middle of a confluence region joined by multiple branches.

Distance transforms can be used to generate skeletons [6, 11, 26]. A set of points, whether they are symmetry points [6] or local maxima plus saddle points [26], are selected and special paths are designed to connect these points for topological invariance. These algorithms are non-iterative and therefore are width-independent, similar to the method of generating median lines based on contours. The original object can be reconstructed from the skeleton exactly or approximately within a specified error. Choosing an appropriate metric, that provides a good approximation to the Euclidean distance, makes the skeleton less sensitive to rotation. The main problem of these algorithms is that in some pathological cases, when two connecting paths are "tangent" to each other, false single-pixel holes can occur in the skeletons and need to be removed by an additional special check.

The Voronoi diagram is another method for computing a connected, regenerative, Euclidean skeleton of an object. However the algorithm to compute the Voronoi diagram is very complex due to the linked graph structure and is quite memory-intensive [10].

A new boundary based thinning algorithm is developed in this paper, which can deal with objects with a general shape. The algorithm can be outlined as follows: Given a proposed shrinking width w_p, two scans are made, first transversely and then vertically, to delete $w_p/2$ pixels of the object on both sides in one scan. If the connectivity is broken, reconnection in an 8-connected fashion is made. The process stops when no further pixels are deleted.

II. BOUNDARY BASED THINNING ALGORITHM

It is assumed that multiple limbs intersect each other either at a point or at a portion of a line in the general case (Fig. 1) and there is no restraint on the shape of objects. For example, suppose the line m is shrunk into one point. If k=2 and j=2, it could be a pair of intersection lines; if k=0 and j=2, it could be a face-down angle; and if k=2 and j=0, it could be a face-up angle.

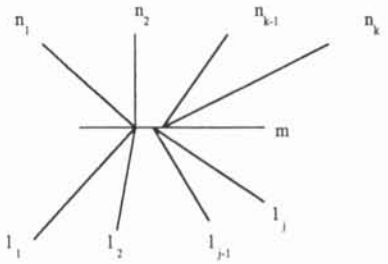


Fig.1 Multiple limbs thinning model

Given a proposed width w_P, first make a transverse scan, row by row, from top left to right bottom. If the width of the object w_o at a row is less than w_P, its center point is retained, labelling the shrunk width of both left and right sides. If w_o is odd, the shrunk width of both sides is the same, equal to $(w_o-1)/2$. If w_o is even, the shrunk width is $(w_o/2)-1$ for the left side and $w_o/2$ for the right side. If $w_o=1$, no point can be deleted and the shrunk width is equal to zero. If $w_o \geq w_P$, $w_P/2$ pixels are deleted on both sides and the new boundary points are labelled with the shrunk width. During the operation, the number of deletions on either side is always recorded. In order to preserve the property of connectivity that the object possesses, it is necessary to compare the sum of the shrunk width of the present row (i) at a point on either side (for example left side) and the shrunk width of the last row (i-1) at a point on the opposite side (in this context it is right side), with the difference of the column numbers of the point in the present row (nL) and the point in the last row (jLp) within the search scope. If the latter is less than or equal to the former, i. e. $(nL-jLp) \leq (L+rp+1)$ it means that the two lines of the object were connected before shrinking, and the shrinkage breaks the original connectivity. The shrunk points should then be recovered in the 8-connected fashion within both the shrunk widths and the labelling values modified. The process only changes object points on the boundary into background points and never changes background points into object points. It deletes as many as possible of the object points on the boundary (within the limitation of the proposed width), without violating the connectivity in each row scan. It is actually a three-line support to secure that points in the middle line (ith row) are connected with points on both the line above ((i-1)th row) and the line beneath ((i+1)th row), if they were originally connected. The process is separated into two steps: First, the connectivity of the points in the middle line and in the line above is kept and then the connectivity of the points in the middle line and in the line beneath is obtained (Fig. 2). From its construction, it is topologically invariant.

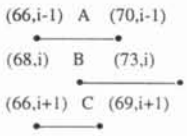
Since more than one line is shrunk, the remaining point may need to keep connectivity with more than one point. It is therefore necessary to detect in the range which is equal to the shrunk width on both sides of a point or a line to keep the connectivity of the object, when multiple limbs intersect each other at a segment of a line.

The process is the same in the vertical scan, column by column. The two cycles repeat until no more pixels of the object can be deleted.

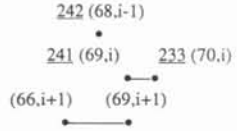
If $w_o=2$, the right side pixel in the transverse scan or the lower pixel in the vertical scan will be deleted and the left side pixel or the upper pixel retained. If the deleted pixel is changed back into the object pixel, because of the consideration of connectivity, the retained pixel is checked to find if it is a simple point [24] using conditions (a) and (b) of [16].

The operation is a shrinking process. It will shrink a vertical line, or a horizontal line into one point, and also a vertical line crossing a horizontal line. This is really not what is wanted. So in the process of checking connectivity, it is necessary to ensure that if a shrunk vertical or horizontal line is unit width it should be restored unless its neighbour line is also shrunk in the scan.

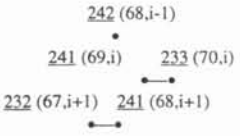
The algorithm shares the idea of the contour generating method (CGT) [2] in using the extra buffers for storing the locations of all boundary points of the objects in the first scans and skipping the points of the background in the further scans. The details can be found in the appendices.



a. Shrink A to the midpoint (68,i-1) and label 242*.



b. Shrink B to the midpoint (70,i) and label 233. In order to preserve connectivity between A and B, release the point (69,i) and label 241, which means there is one point deleted to its left.



c. Shrink C to the midpoint (67,i+1) and label 232, release the point (68,i+1) and label 241, which means there is one point deleted to its right.

Fig. 2 Shrinking process and how it preserves the original connectivity.

* The labelling value between [231,239] means that w_o is even and the left shrunk width is one less than the right one; the number of deleted points is equal to the labelling value minus 230. The labelling value between [240,249] means that w_o is odd and the shrunk width is the same for both sides; the number of deleted points is equal to the labelling value minus 240.

III. DISCUSSION AND EXPERIMENTAL RESULT

The evaluation of a thinning algorithm is to see if it is topological and shape preserving and its sensitivity to the boundary noise. The algorithm proposed deletes the maximum possible $(w_P/2)$ points on one side and preserves the connectivity.

It is difficult for a local mask (k×k, k≥3) to distinguish the boundary noise from the boundary points. Using the

new algorithm boundary noise cleaning can be easily accompanied with the first transverse and vertical scans. The proposed algorithm does not cause excessive erosion in dealing with ±45° slant, horizontal and vertical lines with two pixel width. It can obtain almost minimally 8-connected curves except for the situation shown in Fig. 13 of [6], when different branches converge towards a common position and the deletion of any pixel in the core will break connectivity. For k×k thinning, the maximum possible shift increases as k increases. Spurs may occur for wide lines, large curvatures and large k. The algorithm proposed tends to contract extensively in the horizontal direction because of the sequential order (first transverse then vertical scan). This is so especially when the width of a block w_o is less than w_p; then the block of an object will be transformed into a vertical line even though its thickness is less than its width. Spurs may also occur for large w_p. It is a trade-off between speed and shape preserving when choosing the proposed width w_p.

Suppose the image is an $M \times M$ matrix. For WIFT [6], the total number of inspected pixels is $6M^2+\|K\|$, where K is the set of skeleton pixels of the object. For CGT [2], the total number of inspected pixels is $M^2+\|S\|$, where S is the set of the objects pixels. For the new algorithm, the total number of inspected pixels is $2M^2 + O(\|S\|)$. Since the algorithm inspects not only the boundary pixels but also the inner pixels during the check of the connectivity, the running time is proportional to $(2/ w_p) \times \max w_o$.

The algorithm significantly reduces the number of iterations compared to other iterative algorithms. In existing sequential thinning algorithms, contour generating methods are faster. Although XW's [2] and WZ's [3] algorithms are similar in inspecting pixels along the boundaries rather than all pixels in each iteration; XW's algorithm is simple in producing a good quality skeleton. The two algorithms were coded in Borland C++ and run on a PC 386 SX-160.

The results of the timing test for thinning patterns "A", "leaf", "bull's eye" and Chinese character " " with 128× 128 pixels (Fig. 3) is given in Table 1. The times listed do not include the time of input and output of files as this time is the same for all algorithms. The number of iterations for XW's algorithm may make less sense since its running time is actually proportional to the total number of the object pixels. If the width of the object is small, XW's algorithm may be faster than the new algorithm, because the basic cost of it is larger than that of XW's algorithm as mentioned above. The larger the width of the object to be thinned, the faster the new algorithm becomes relative to XW's.

IV. CONCLUSION

A new boundary based thinning algorithm is developed. It has a broader scope, from a local area to a global area, and significantly reduces the number of iterations compared with existing algorithm. A thinning operation model, which is applicable to general shapes of objects, is introduced. It is topologically invariant from its construction. It is also less sensitive to boundary noise and obtain the almost minimally 8-connected skeleton. If the proposed width is large, the algorithm will cause the skeleton to contract in one direction and spurs to occur, but the distortion is limited in range. The computation complexity is modest and it is shown that with an increase in the width of the object to be thinned, for example, with increase of the resolution of digitization, the proposed algorithm is computationally efficient.

a. English alphabet "A" b. "leaf"

c. "bull's eye" d. Chinese character " "

Fig. 3 Thinning patterns

APPENDIX I

List of Symbols :

TBUF: array for storing the coordinates of boundary points in the transverse scan.

VBUF: array for storing the coordinates of boundary points in the vertical scan.

w_o: the width of the object.

w_p: the proposed shrinking width.

nL: the column value of the leftmost point of the central part remaining after shrinking in the ith row.

nr: the column value of the rightmost point of the central part remaining after shrinking in the ith row.

(If $w_o \le w_p$ the central part is shrunk into one point and nL=nr.)

jLp: the column value of the up left point in the $(i-1)$th row within the search scope.

jrp: the column value of the up right point in the $(i-1)$th row within the search scope.

L: labelling value, which means L points on the left have been deleted in the ith row from the leftmost point of the central part.

r: labelling value, which means r points on the right have been deleted in the ith row from the rightmost point of the central part.

Lp: labelling value, which means Lp points on the left have been deleted in the $(i-1)$th row from the leftmost point of the central part.

Table 1. Comparison of run time (in seconds)

Algorithm	pattern							
	A		leaf		bull's eye			
	no.	time	no.	time	no.	time	no.	time
New algorithm	4*	0.384615	7	0.714286	3	0.604396	5	0.549451
Xu & Wang	9	0.494505	24	0.824176	6	0.934066	10	0.769231

* number of the iteration. The proposed width is 8.

rp: labelling value, which means rp points on the right have been deleted in the $(i\text{-}1)$th row from the rightmost point of the central part.

[nL-L-w_p/2, nL]: the left side search scope.

[nr, nr+r+w_p/2]: the right side search scope.

APPENDIX II

Algorithm:

1. Initialize.

2. Put the coordinates of the boundary pixels into TBUF from top to bottom in the transverse scan of the image.

a) Retain the central part of the line segment, delete the shrunk points of both sides and label the shrunk width of both sides at the leftmost point and the rightmost point of the central part. (If $w_o \geq w_p$, w_p/2 points are deleted from both sides. If $w_o < w_p$ and if w_o is odd $(w_o\text{-}1)/2$ points are deleted from both sides. If w_o is even $(w_o/2)$-1 points are deleted from the left side and $w_o/2$ points deleted from the right side. If w_o=1, no point is deleted from either side.)

b) Check the connectivity of horizontal lines. Within the left side search scope [nL-L-w_p/2, nL], check the connectivity--if (nL-jLp[t])≤(L+rp[t]+1) (t=0, ·····k_l), restore the connection within both shrunk widths (for the ith row it is L; for the $(i\text{-}1)$th row it is rpt[t]) and modify the label values of the released points. Within the right side search scope [nr, nr+r+w_p/2], check the connectivity--if (jrp[t]-nr)≤(r+Lp[t]+1) (t=0, ····· k_r) do the same as for the left side. If w_o=2, the right side point is changed back into the object point, and a check of a simple point for the remained point is made.

3. In the vertical scan of the image, the coordinates of the boundary pixels are put into VBUF from left to right.

The next processes are similar to subprocesses a) and b) in the transverse scan .

4. Update the content of the boundary point buffers after each scan and repeat the steps 2 and 3.

5. The whole process stops when no further point can be deleted.

REFERENCE

1. N. J. Naccache and R. Shinghal, "SPTA: A proposed algorithm for thinning binary patterns," *IEEE Trans. Syst., Man, Cybern.* **SMC-14**, pp. 409-418, 1984.

2. Wen Xu and Chengxun Wang, "CGT: A fast thinning algorithm implemented on a sequential computer," *IEEE Trans. Syst., Man, Cybern.*, **SMC-17**, pp. 847-851, 1987.

3. P. S. P. Wang and Y. Y. Zhang, "A fast and flexible thinning algorithm," *IEEE Trans. Comput.* **38**, pp. 741-745, 1989.

4. T. Pavlidis, "A thinning algorithm for discrete binary images," *Comput. Graphics Image Processing*, **13**, pp. 142-157, 1980.

5. ____, "An asynchronous thinning algorithm," *Comput. Graphics Image Processing*, **20**, pp. 133-157, 1982.

6. C. Arcelli and G. Sanniti di Baja, "A width-independent fast thinning algorithm," *IEEE Trans. Pattern Anal. Math. Intelligence* **PAMI-7**, pp. 463-474 , 1985.

7. C. Arcelli, "Pattern thinning by contour tracing," *Comput. Graphics Image Processing*, **17**, pp. 130-144, 1981.

8. M. Pilar Martinez-Perez, Javier Jimenez and Jose L. Navalon, "A thinning algorithm based contours," *Comput. Vision Graphics Image Processing*, **39**, pp. 186-201, 1987.

9. B. Saphiro, J. Pisa, and J. Sklansky, "Skeleton generation from x, y boundary sequences," *Comput. Graphics Image Processing*, **15**, pp. 136-153, 1981.

10. J. W. Brandt and V. R. Algazi, "Continuous skeleton computation by Voronoi diagram," *CVGIP: Image Understanding*, **55**, pp. 329-338, 1992.

11. C. Arcelli, L.P. Cordella, and S. Levialdi, "From local maxima to connected skeleton," *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-3**, pp. 134-143, 1981.

12. A. Rosenfeld, "A characterization of parallel thinning algorithms," *Inform. Control*, **29**, pp. 286-291, 1975.

13. R. Stefanelli and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," *J. Assoc. Comput. Math.* **18**, pp. 255-264, 1971.

14. A. Rosehfeld and L. S. Davis, "A note on thinning," *IEEE Trans. Syst., Man, Cybern.* **SMC-6**, pp. 226-228, 1976.

15. H. Tamura, "A comparison of line thinning algorithms from digital geometry viewpoint," *Proc. 4th Int. Conf. Pattern Recognition*, pp. 715-719, 1978.

16. T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Comm. ACM* **27**, pp. 236-239, 1984.

17. H. E. Lu and P. S. P. Wang, "A comment on 'A fast parallel algorithm for thinning digital patterns," *Comm. ACM* **29**, pp. 239-242, 1986.

18. C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott, "An improved parallel thinning algorithm," *Comm. ACM* **30**, pp. 156-160, 1987.

19. R. T. Chin, H. K. Wan, D. L. Stover and R. D. Iverson, "A one-pass thinning algorithm and its parallel implementation," *Comput. Vision Graphics Image Processing*. **40**, pp. 30-40, 1987.

20. R. W. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation," *Comm. ACM* **32**, pp. 124-131, 1989.

21. Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Comm. ACM* **32**, pp. 359-373, 1989.

22. Z. Guo and R. H. Hall, "Fast fully parallel thinning algorithms," *CVGIP: Image Understanding*, **55**, pp. 317-328, 1992.

23. L. O'Gorman, "k×k thinning," *Comput. Vision Graphics Image Processing*, **51**, pp. 195-215, 1990.

24. T. Y. Kong and A. Rosenfeld, "Digital topology: Introduction and survey," *Comput. Vision Graphics Image Processing*. **48**, pp. 357-393, 1989.

25. L. Lam, S-W Lee and C. Y. Suen, "Thinning methodologies -- A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intelligence*, **PAMI-14**, 869-885, 1992.

26. C. W. Niblack, P. H. Gibbons and D. W. Capson, "Generating skeletons and centerlines from the distance transform," *CVGIP: Graphical Models and Image Processing*, **54**, pp. 420-437, 1992.